Software Design Document (SDD) Template

Software design is a process by which the software requirements are translated  into a representation of software components, interfaces, and data necessary for  the implementation phase. The SDD shows how the software system will be  structured  to  satisfy  the  requirements.  It  is  the  primary  reference  for  code  development and, therefore, it must contain all the information required by a  programmer to write code. The SDD is performed in two stages. The first is a  preliminary design in which the overall system architecture and data architecture  is defined. In the second stage, i.e. the detailed design stage, more detailed data  structures are defined and algorithms are developed for the defined architecture.

This template is an annotated outline for a software design document adapted  from the IEEE Recommended Practice for Software Design Descriptions. The  IEEE  Recommended  Practice  for  Software  Design  Descriptions  have  been  reduced  in  order  to  simplify  this  assignment  while  still  retaining  the  main  components and providing a general idea of a project definition report. For your  own  information,  please  refer  to  IEEE  Std  1016­19981  for  the  full  IEEE  Recommended Practice for Software Design Descriptions.

**“Informatic distributed system for managing the activities of an association of tenants”**

**Software Design Document**

Name (s):  **Prepared by Calina Florin, Marcu Andrei, Matei Alexandru, Tolgoi Dragos**

Date: 03/24/2020

**TABLE OF CONTENTS**

**1.  INTRODUCTION**

1.1  Purpose

1.2  Scope

1.3  Overview

1.4  Reference Material

1.5  Definitions and Acronyms

**2.  SYSTE MOVERVIEW**

**3.  SYSTEM ARCHITECTURE**

  3.1  Architectural Design

  3.2  Decomposition Description

  3.3  Design Rationale

**4.  DATA DESIGN  3**

4.1  Data Description  3

4.2  Data Dictionary  3

**5.  COMPONENT DESIGN**

**6.  HUMAN INTERFACE DESIGN**

6.1  Overview of User Interface

  6.2  Screen Images

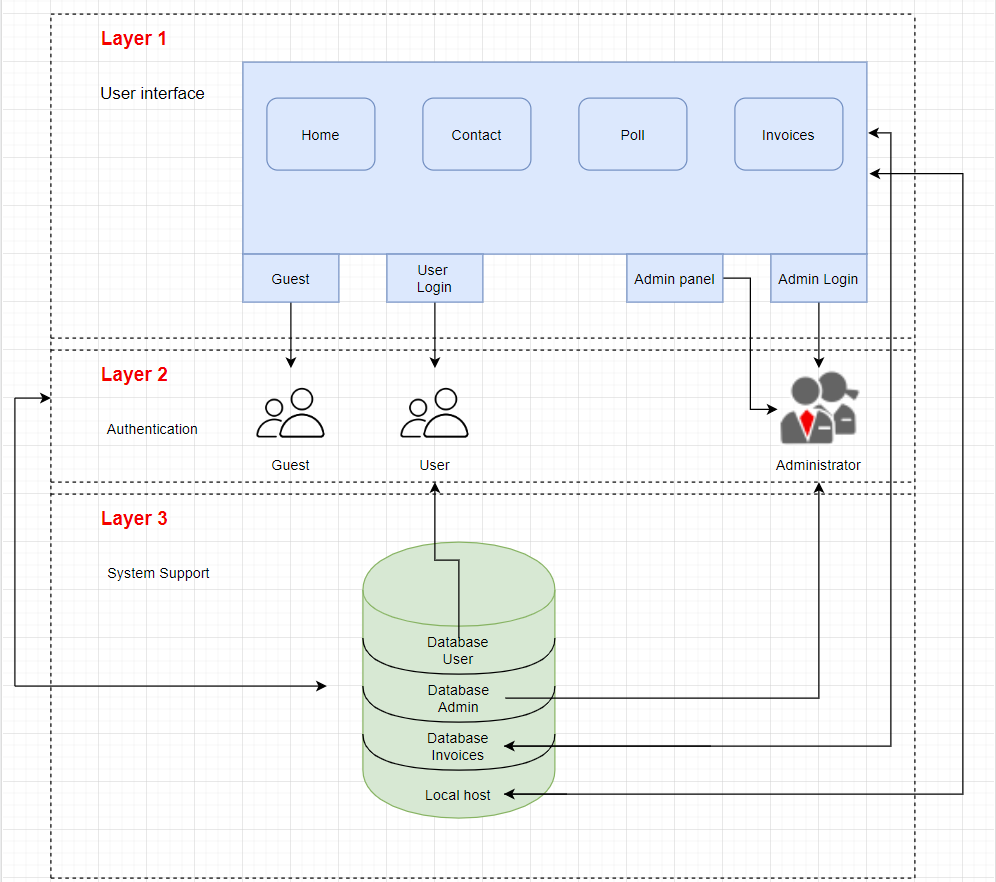
6.3  Screen Objects and Actions

**7.  REQUIREMENTSMATRIX**

**2.  SYSTEM OVERVIEW**

This document describes the requirements for handling the activities of an association of tenants. The system will have a user interface, a database server which will store information about the activity of the association application server. This software system will be a web application for an association of tenants. This system will be designed to ease the administrators’ management activities and also help the tenants to easily have acces to all their invoices, see how much they have to pay, see their overdue invoices and anything related to their building services.There will be two different user interfaces that will accompany this website: one for the users and one for the administrators. The administrators will be able to add users and remove them, to add bills, to create pools etc. The users will be able to see the bills, answer to the pools created by the administrators etc.

**3.  SYSTEM ARCHITECTURE**

**** **3.1  Architectural Design**

**Layer communication**

Layer 1

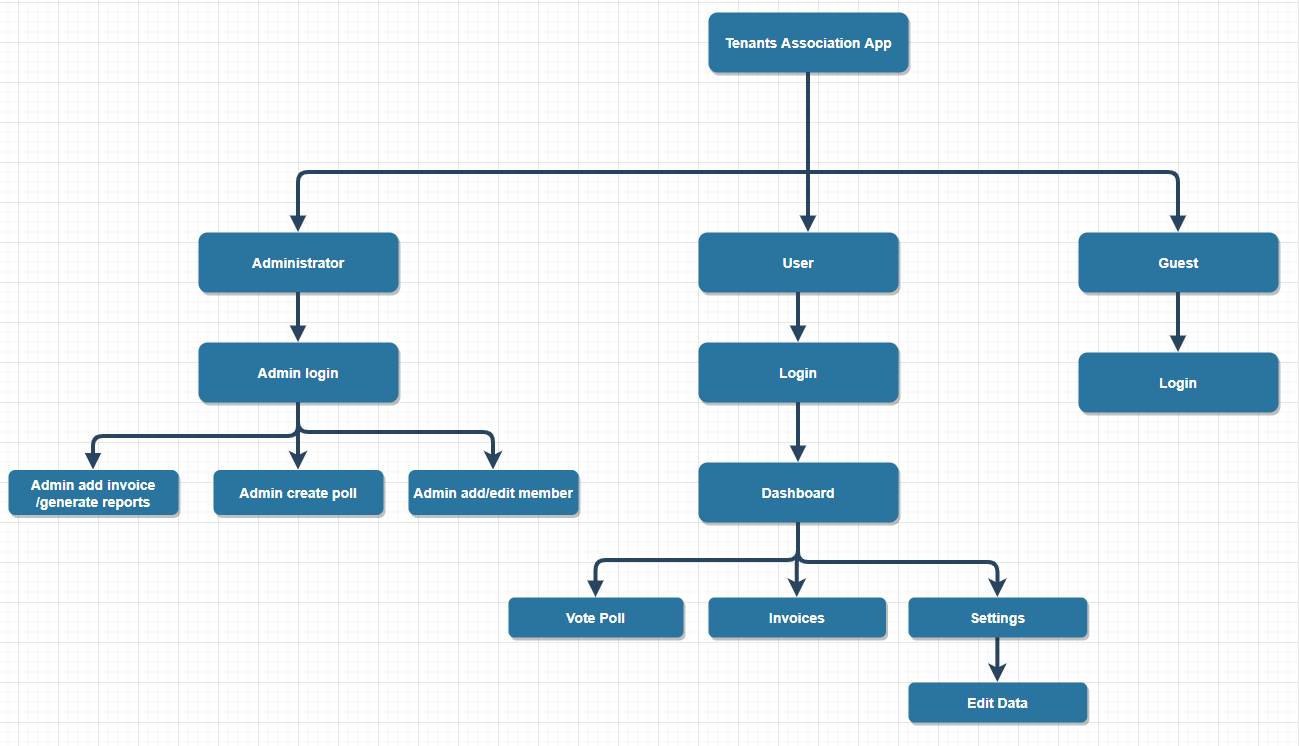
* Guest can only acces the login page.The app is only inteded for a specific target audience.
* Users can log in and have acces to all the application features:view their invoices/overdue payments/vote in a poll/etc.
* Administrator can login on the website and manage the users(add/remove),add an invoice/generate reports/add poll/etc.

Layer 2 to layer 3

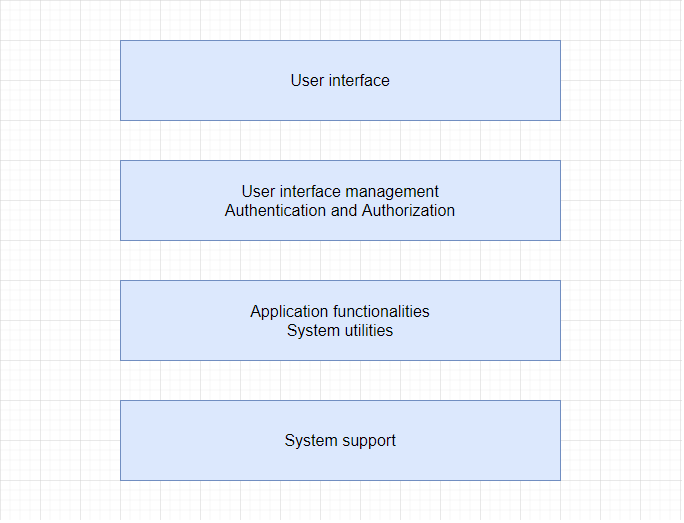
* Guests have no relation.
* When users will login on the website will cause a querry to be sent to the database.When a user will modify his data(ex: nr. of tenants) these changes get added to the database.
* Administrators when will login will cause a query to be sent to the database.When the administrator will add/remove users/invoices, the changes will get added to the database.

Layer 1 to layer 3

* Data inserted by the users or the administrator may change the appearence of the website.

**3.2  Decomposition Description**

**3.3  Design Rationale**

****

Database

There are four layers in this architecture where each layer has a connection between modularity and component within them. From top to bottom, they are:

* The presentation layer : It contains all categories related to the presentation layer(U.I).
* The business layer : It contains business logic.(ex: Login)
* The persistence layer : It’s used for handling functions (application functionalities)
* The database layer : This is where all the data is stored.

**Advantages**

* Each layer can be assigned to a different part of the team
* It can be used when there is a need for multi-level security
* All ‘similar’ components are together and the other reason is that it provides layers of isolation.

**Disadvantages**

* Layers are closed, meaning a request must go through all layers from top to bottom
* Performance can be a problem because a request is processed at each layer.